Personal draft of the article published as:
Sobreira, P., Tchounikine, P. (2015). Table-Based Representations Can be Used to Offer Easy-to-use, Flexible, and Adaptable Learning Scenario Editors. Computers & Education 80, p. 15-27. DOI: 10.1016/j.compedu.2014.08.002

Table-Based Representations Can be Used to Offer Easy-to-use, Flexible, and Adaptable Learning Scenario Editors

Abstract

This article presents an innovative approach to the design of learning scenario editors for teachers as not-specifically-trained users. The approach features simplicity, flexibility, and easy adaptation to local contexts and needs. It is based on the governing design decision to provide a basic representation device known to be easy to use and adopted by teachers in standard practice: a table, similar to that found in office suites. The table is put in structural correspondence with a pivotal model (a tree), which is used to implement services requiring complex mechanisms that can be found in state-of-the-art systems, for example instantiation support or operationalization. We show how this approach makes it possible to design simple, flexible editors that offer complex services as add-ons and are easy to adapt to users' local needs, practices, and/or technical ecosystems.

1. INTRODUCTION

This article presents an innovative design rationale and technical approach to the issue of offering teachers easy-to-use, flexible, and adaptable learning scenario editors (in this article, "teacher" is used as an inclusive term for K-12 teachers, university lecturers or vocational/professional trainers).

A learning scenario is a learning design that specifies a particular arrangement of tasks that students must engage in during a session, and identifies aspects such as the associated resources or roles (as examples: role play, predict-observe-explain, jigsaw or pyramid (LAMS, 2013; Hernández-Leo et al., 2010)). Research related to learning scenarios takes place in the field of Learning Design, which is defined in the Larnaca Declaration on Learning Design as "[the field that] seeks to develop a descriptive framework for teaching and learning activities ("educational notation"), and to explore how this framework can assist teachers to share and adopt great teaching ideas" (Larnaca, 2012). Scenarios are one type of learning unit, which differs from others such as courses.

A learning scenario editor (LSE) is a tool for representing and editing learning scenarios. LSEs provide an editing interface as a basic service. Depending on the system, they may also provide other services that we will refer to as "additional services", such as support for instantiating scenario patterns or operationalization of the scenario on enactment frameworks like learning management systems (LMSs).

Different works and collective syntheses have highlighted that it is desirable for teachers to be able to engage with modeling tools to participate in authoring processes, share constructions, and disseminate good practices (Larnaca, 2012; Derntl et al., 2011; Tchounikine et. al., 2009). However, to date, modeling languages and LSEs have not been widely adopted by either practitioners or institutions (Derntl et al., 2011).

Many reasons explain why teachers do not commonly use LSEs, including teachers' technical skills and professional practices, institutional aspects, and school equipment (Larnaca, 2012; Mor and Craft, 2012; Demetriadis et al., 2003). One of these reasons is related to LSEs as computer-based systems. Research has highlighted that using representations inspired by data or process modeling, such as XML-like trees or process charts, may be an issue for adoption as these representations are not commonly used by teachers (Neumann et al., 2010). The problem does not seem to be that teachers cannot understand complex modeling or meta-modeling processes: studies have shown that teachers can handle complexity (Derntl et al., 2011). However, teachers aim to manage their teaching setting, not to become modeling specialists. Instructional designers act as professional modelers, tend to have specific training, and are interested in the means for specifying scenarios or indexing them; in direct contrast, teachers generally do not have any specific modeling training and are mainly interested in how to adapt/instantiate scenarios and deliver them in real classrooms (Weinberger et al., 2009). To be used by teachers, LSEs must be simple, intuitive, and flexible (Dillenbourg and Jermann, 2010). This conclusion concurs with Delphi studies showing that teachers generally believe that software should be simple, easy to learn, and adaptable (Williams et al., 2004).

Designing simple, flexible and adaptable LSEs is a challenge. To be easy for teachers to use, the LSE interface must lie within their representation and technical skills. To be easily adaptable, the LSE must have been natively designed as computationally flexible. Unfortunately, implementing additional services often leads to complex interfaces and/or complex software architectures and, as a result, to low adaptability.

This article presents an approach based on an alternative to traditional design rationales (see Section 3). It is based on the governing design decision to provide a basic representation device known to be easy to use and adopted by teachers in standard practice: a table, similar to that found in office suites. Technically, this table representation is put into structural correspondence with a tree representation (which is hidden to users). The tree representation is used to implement the additional services that can be found in LSEs, such as instantiation support or operationalization. Under this design strategy the complex representations or manipulations requested by additional services do not hinder the representation simplicity. We show how this approach makes it possible to represent and operationalize a wide range of scenarios while providing teachers a system that is easy to use and easily adaptable to local contexts and needs. This approach is not meant to replace more comprehensive editors but to enrich the existing technical offering with LSEs that feature simplicity, flexibility, and adaptability, and that can be used when sufficient and convenient (see Section 9).

The methodology for this work is as follows. The contribution is an approach that allows designing easy-to-use and adaptable LSEs. To study this output, four objectives/evaluation criteria were set: (1) pedagogical expressiveness, i.e., can table-based editors represent a wide range of scenarios?; (2) usability, i.e., do teachers find the editor easy to use and intuitive?; (3) computational expressiveness, i.e., does the approach allow for implementation of additional complex services?; and (4) computational flexibility, i.e., is the editor easy to adapt to local needs? The general requirements and services to be offered were identified by examining state-of-the-art LSEs. Table pedagogical expressiveness was tested by studying how the editor handled representation of different types of scenario. Ease-of-use was evaluated with usability tests. Computational expressiveness and computational flexibility were demonstrated by developing proofs-of-concept. (As usual when considering such editors, the quality of the learning designs produced by teachers during the tests is not considered, as good and bad designs may be produced using any editor.)

The rest of the article is organized as follows. In Section 2, we present a short review of LSE services, i.e., of the general requirements. In Section 3, we present the adopted design rationale. In Section 4, we present the proposed approach (the table/tree model). Sections 5-8 address the different objectives and evaluation criteria: pedagogical expressiveness, usability, computational expressiveness, and computational flexibility. Finally, we discuss the approach in Section 9 and conclude in Section 10. Throughout the article we use different examples of instances of the editor we designed. Use cases, other examples, and technical aspects are presented in more detail in the supplementary material.

2. REPRESENTING AND MANAGING LEARNING SCENARIOS

Representing and managing learning scenarios involves different activities. These activities may be conceptualized and articulated in multiple ways, and involve various players, but a review is out of the scope of this article. In this section, we sketch a general perspective and list some concerns and services described in literature from the field.

2.1. Review of activities

Design is the identification of the scenario's basic principles, i.e., how the featured tasks are supposed to lead to the intended learning goals, and important conditions that must be met. Depending on the context or institution, actors may be instructional designers and/or teachers. As an example, the predict-observe-explain scenario (LAMS, 2013) makes students engage in the following sequence of tasks: (1) observe and describe what they see; (2) provide an explanation for what they have observed and how it relates to their prediction; and (3) explore why this prediction was wrong, if such was the case. As another example, the collaborative jigsaw scenario works as follows: (1) "expert groups" of students are created and provided with resources related to a given topic, the goal being to make them experts in different subjects; and (2) "jigsaw groups" comprising experts in different subjects are created and asked to accomplish a task requiring all subjects, with the expectation that students with different expertise will engage in knowledge-generative interactions such as explaining things to each other, debating, or solving conflicts. Representations of this jigsaw scenario as a table and as a workflow can be seen in Figure 2 and in the supplementary material, respectively.

Editing is the adaptation and/or instantiation of the scenario to the subject being taught, the teaching context, and the real students. It is performed by the teachers. For instance, using the jigsaw scenario to have students address energy-saving issues requires instantiation and editing actions, such as deciding that expert groups will focus on "insulation" and "heating", deciding how many expert groups will be created and dividing the students, deciding how the students in the expert groups will be mixed to create the jigsaw groups, and defining the resources provided to students during each phase (e.g., online documents during the expert phase and a quiz during the jigsaw phase). When editing scenarios, teachers must keep in mind both the scenario design rationale (e.g., creating jigsaw groups) and the local and contextual issues (e.g., number and characteristics of students or difficulty of the actual tasks). This may lead to adaptations (e.g., add intermediate activity). Editing is the main activity considered in this paper. Teachers' basic practices generally consist in reusing, adapting, and instantiating existing scenarios designed by instructional designers.

Operationalization refers to deployment of the design on the technical framework provided to students to perform the scenario tasks. Enactment platforms include modeling language players (e.g., Coppercore (2012) or LAMS (2012)), semi-specific architectures related to different types of scenario (e.g., CeLS (Ronen et al., 2006) for collaborative scenarios), and specific architectures (e.g., jigsaw-specific architecture implementing specific mechanisms (Dillenbourg and Hong, 2008)). In actual practice, general frameworks (e.g., wikis) and LMSs (e.g., Moodle (2012)) are frequently used. Operationalization is not always considered because LSEs may sometimes be used only to prepare sessions.

Orchestration (Dillenbourg et al., 2012; Tchounikine, 2013) refers to management of the session. While students enact the scenario, teachers must conduct the session (e.g., clarify instructions or provide support) and/or manage events requiring run-time adaptation of the scenario (e.g., add a task or move a student from one group to another).

2.2. Review of services and characteristics

Representation. The basic service provided by LSEs is offering a notation system and means of manipulation that make it possible to represent, instantiate, and adapt the scenario (design, editing, orchestration). When considering LSEs as a support for thinking about and analyzing options, it is clear that editing (e.g., adding or removing tasks or changing resource allocation) must be easy to perform. In terms of representation power, most scenarios are simple and can be represented as a list of phases or tasks to be completed, with by-extension listing of students or resources. However, some scenarios require *representation of complex constructions*, such as complex scheduling (e.g., branching or loops), by-intension mechanisms (e.g., "G3 group is made up by mixing G1 and G2 students who have not yet worked together"), or run-time dynamic principles (e.g., "the set of students who answered quiz Q1 correctly"). With respect to adoption, an important aspect is *representation flexibility*, i.e., the possibility for users to adapt the representation structure. Editors provide support by introducing conceptual notions. However, this imposes a particular way of conceptualizing and representing scenarios, which may correspond poorly to the perspectives of individual teachers. For instance, although the end result may be a similar operationalization, conceptualizing a scenario as "a set of activities to be performed by groups sharing some resources" is different than seeing it as "a set of participants provided with the resources associated with activities". Offering teachers the flexibility to adopt the approach they prefer is important in contexts in which adoption is positively affected by whether teachers can stick to their practices or perspectives.

Instantiation support. Scenarios are usually defined as abstract patterns featuring pedagogical principles. Instantiating a pattern is easy when limited to a few items but may become tricky as more items are involved. Studies have shown the value of supporting teachers in dealing with certain aspects of instantiation (e.g., for a jigsaw scenario, preparing group division before the class or adapting the groups to unexpected situations) when there are many constraints to control and a large number of students (Pérez-Sanagustín et al., 2009; Villasclaras-Fernández et al., 2009). Another instantiation issue for teachers relates to considering both their contextual constraints and the scenario's intrinsic constraints (Dillenbourg and Tchounikine, 2007). This may be resolved by implementing *constraint-checking mechanisms*. As an example: in a jigsaw scenario, if a student is moved from one group to another, check that the jigsaw groups are still made up of students who were in different expert groups.

Operationalization. Some LSEs offer a scenario-oriented interface from which the enactment framework may be configured. With respect to operationalization, some authors have argued that *simulation* (testing the scenario before introducing it to students) may provide useful input for analyzing and fine-tuning the scenario (Harrer et al., 2007).

3. DESIGN RATIONALE

In order to explain the design rationale of the approach we studied, we will differentiate it from that of other works addressing Learning Design editors (which may be specific to learning scenarios or more general).

3.1. Traditional design rationales and design methodologies

In general, Learning Design editors are designed as means to make a given educational modeling language (EML) easy to use for non-computer scientists (Botturi and Stubbs, 2007). An EML is a machine-interpretable semantic notation for a more or less broad variety of designs. EMLs are designed with specific objectives. The prototypical example is IMS-LD (IMS-LD, 2013), which aims to be comprehensive and achieve standardization and industrialization. For this purpose, IMS-LD includes notions that are supposed to cover a large spectrum: activity, role, environment, learning object, learning objective, prerequisite, and service. The way these notions may be used to model units of learning is defined by the language meta-model. IMS-LD editors such as ReCourse (2013) are designed to make this specification easy to use. They make it possible to create IMS-LD-compliant units of learning, including learning scenarios.

Following the same line of thinking, many LSEs are based on a model that defines what a scenario is, i.e., on a set of conceptual notions and their relations. SCY-SE (Lejeune et al., 2009) focuses on inquiry learning scenarios and builds on a model featuring notions such as learning activity space, emerging learning object, tool, activity, scaffold, and resource. MoCoLADe (Harrer et al., 2007) and various other editors focusing on computer-supported collaborative learning (CSCL) scenarios reuse a model featuring the activity, group, participant, role, and resource notions (Kobbe et al., 2007). Other examples may be found in (Botturi et al., 2006), (Larnaca, 2012), and (Prieto et al., 2013). Some of

these editors, such as LAMS (2012) and CeLS (Ronen et al., 2006), closely associate teacher authoring and learner implementation of activities on a specific enactment platform. In such cases, operationalization is straightforward.

A different design rationale is to offer scenario patterns (Goodyear and Retalis, 2010) that users can instantiate, such as Collage editor (Hernández-Leo et al., 2006, 2010). The jigsaw pattern is a prototypical example.

Finally, editors may also be designed from a specific contextual requirement analysis. As an example, Learning Designer (a tool for designing conventional, digital, and blended learning) targets academics as "not trained teachers (with) little time or support to work on innovation and improvement of their teaching" (Laurillard et al., 2013). The design involved ten "informant practitioners" to identify practitioners' conceptions of learning design. The requirement analysis provided the basis for developing a well-informed design.

3.2. Exploring another perspective

A common characteristic of the approaches explored above is that the editor is designed to make a given representation easy to use. Simplicity and flexibility are addressed via usability studies and/or participative design. Technical adaptability of the editor is not considered: the editor and its associated services are specified only once.

If simplicity and flexibility are considered primary concerns, this approach has some limits. First, when considering an existing EML, the LSE interface must comply with this language characteristics and constraints. Second, with respect to participative design, gathering a representative set of informant practitioners may be a challenge. A known issue is that teachers' actual use of technologies is related to a variety of contextual factors, for example their local needs, personal professional practice, and local technical ecosystem. Unless targeting very precise contexts and groups of users, it is difficult to anticipate usages. Moreover, usages evolve.

An alternative way to consider simplicity and flexibility involves adopting a Copernican revolution: build from an interface that is known to be simple and adopted by target users as standard practice. We considered tables. Then, following this line of thinking, make it possible to adapt the editor to users' contextual needs by introducing representation flexibility (i.e., allowing users to adopt different representations) and computational flexibility (i.e., implementing the editor in a way that enables easy adaptation to address specific needs). Such an approach has intrinsic limits, e.g., complex constructions cannot be straightforwardly represented by a basic table and so require add-ons. It is therefore not meant to replace other approaches but simple to expand the offering of modeling tools available to include an editor that natively addresses simplicity and eases transition from existing practices, which can be used when sufficient and convenient (see discussion in Section 9).

Building an LSE from a table representation raises different questions: Does a table, though not comprehensive, allow teachers to represent a large range of scenarios? Do teachers confirm intuitiveness and simplicity? Is a table representation compatible with offering the additional services that can be found in state-of-the-art systems, e.g., representation of complex scheduling, instantiation support, constraint checking, or operationalization? Finally, from a technical perspective, can this approach be implemented in a way that is simple enough to allow adaptability to local contexts at low engineering cost? In Section 4, we present how we implemented this strategy and, in sections 5-8, how these different questions were answered. In Section 9, we will return to a discussion about this design rationale.

4. THE TABLE/TREE PIVOTAL MODEL APPROACH

4.1. Principles: The abstract model

The representation we studied offers an editing interface similar to that of a table editor in an office suite. Tables are known to be simple and effective representation devices that help users shape a representation of their problems (Nardi & Zarmer, 1993). An Internet search to collect examples of pre-structure sheets available (in our national context) to school teachers to edit and prepare their sessions showed that these sheets are almost always tables. Investigations of school teachers' ICT knowledge also shows that using tables is within teachers' standard practice (Trimmer 2006).

The implementation we studied uses a tree as a pivotal model. A scenario is represented as a table in which the columns represent the notions of the conceptual framework used. The table is maintained in structural correspondence with the tree: rows correspond to branches and columns correspond to levels (with an additional fictive root node; see Figure 1). The ediT2 editor we developed following these principles is just one implementation of these principles.

4.2. Examples: Defining an editor by using specific conceptual notions

To use the table/tree model (i.e., instantiating ediT2 with a particular conceptual framework), the notions to be used must be defined. We will use two main examples: an instance with classical collaborative notions (CSCL example) and an ad hoc instance (wiki) that shows how the approach may be used to define an LSE adapted to local needs.

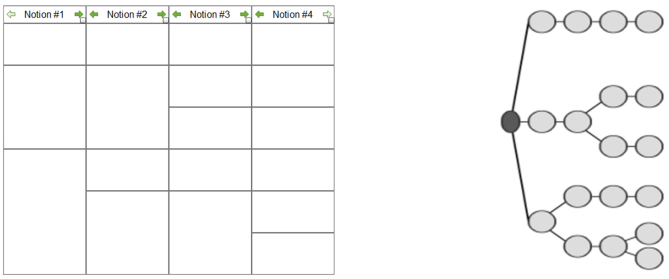

Figure 1. The table/tree correspondence

The CSCL example (Figure 2, left side) illustrates how the editor may be used to enrich the LSE offering given a specific conceptual framework. This instantiation reuses the activity, group, participant, role, and resource notions, which have been identified by consensus (Kobbe et al., 2007). It offers a basic alternative to other LSEs that use these notions, e.g. MoCoLADe and its graph-based representation (Harrer et al., 2007). The wiki example (Figure 2, right side) illustrates how the editor may be used to define ad hoc LSEs dedicated to a particular type of scenario and/or setting. In this case, we considered the following type of scenario: (1) a list of topics is defined, each of which is associated with a synthesis report and a list of sections; (2) students are assigned one or more reports to write and given some input in the form of URLs; (3) the scenario is then enacted via a wiki. To represent such scenarios, the editor was instantiated with the topic, section, editor, and input notions.

Activity	Group	Participant	Resource
Read the general text		s1 s2 s3 s4	General text (in)
Identify techniques	G1	s1 s2	Insulation text (in) Insulation list (out)
	G2	s3 s4	Heater text (in) Heater list (out)
Crossing groups	G1	s1 s4	Insulation questions (in) Insulation list (out)
	G2	s2	Heater questions (in)

Topic	Editor1	Section	Editor2	Input
Brazil's geography	John	Capital	Debbie	URL1
		States	Jim	URL2
Brazil's history	Debbie	Colonization	Bill	URL3 URL4
		Independence	John	URL5
		Early republic	Connie Marge	URL6

The items, e.g., the activities or the editors, can be defined via editing boxes (not shown here). Adding an activity or moving students from one cell to another is natively made easy by the table structure (drag-and-drop manipulations). Rows can be split into sub-rows or merged.

Figure 2. Examples: A CSCL scenario instantiation and a wiki-based scenario instantiation

4.3. General representation principles

The table introduces basic structural "is associated with" semantics that can be interpreted differently according to the notions used. In the jigsaw example, activities "are achieved by" groups, groups "are composed of" participants, and participants "access" resources. In the wiki example, a topic "is associated with" one or more editors and one or more sections, and each section "is associated with" one or more editors and some input.

Additional semantic constraints can be defined by specific pieces of code attached to the table manipulation features. The natively-implemented basic constraint is that an item defined as a notion *N* (e.g., *N*="participant") can only be dropped into the corresponding column of the table. This piece of the code may be enhanced by adding specific *if/then* constraints. This approach makes it possible to implement general controls similar to the ones that may be found in other LSEs (e.g., checking that groups are composed of several participants) and ad hoc controls such as, "*If* a student is the topic editor *then* he/she cannot be an editor for any section of this topic." It also allows for methodological support to be implemented, for example offering a predefined list of activities associated with detailed information (e.g., difficulty or requested level of engagement) and introducing controls to help teachers manage these considerations (e.g., to prevent a particular series of activities or unsuitable division of students); see also Section 8.

The table representation has two intrinsic limitations. First, it only supports simple associations. Second, a table introduces a sequential representation, which means branching constructions or parallelism can only be defined implicitly, e.g., by associating different students with different lists of activities as the teacher did in the jigsaw scenario presented in Figure 2. This is sufficient for simple cases but not for complex constructions. In our implementation, we introduced the additional constraint of 1-n where n $\geq$1 relationships (in other words, a cell has only one parent). This is to allow the table/tree correspondence and facilitate the manipulation of scenarios (analysis, generation, transformation). At the interface level this means that in certain cases, although two sub-rows may have the same values, the corresponding cells cannot be merged and the items must be duplicated. As items may be easily copy-pasted, this is of little consequence for users and was not identified as an issue during usability tests.

4.4. Representation flexibility

A specific expressiveness feature introduced by a table is suggesting/allowing the order of columns to be changed, i.e., considering the notions in different orders. In the interface, columns may be ordered using the left/right arrows at the top of them (see Figure 2). In the tree model, this corresponds to changing the structure of the branches.

Allowing the representation structure to be changed by adding, deleting, or moving a column makes it possible to offer representation variants. For instance, in the wiki example, the "Topic-Editor-Section-Editor-Input" structure suggests that each topic has one or more editors and sections and each section has one or more editors and inputs. A "Topic-Section-Editor-Input" structure suggests that only sections can be edited (no editors for topics). A "Topic-Input-Editor-Section-Editor" structure suggests that the input provided is common to the topic (rather than local to the sections). In fact, the table structure allows for a series of different meaningful structures (in formal ways: [Topic] [Editor]* [Input]* ([Section] [Editor])* || [Topic] [Editor]* ([Section] [Editor])* [Input]*). The table representation makes it easy to use one structure or another and, if the choice is left open, to go from one to the other while modeling.

Different column orders do not necessarily introduce semantic differences, but may still correspond to different ways of conceptualizing a setting. For instance, in one experiment, some teachers used an Activity-Group-Resource representation structure (i.e., represented the scenario as "a set of activities to be performed by groups sharing some resources") while others used a Participant-Resource-Activity (i.e., represented the scenario as "a set of participants given some resources to perform activities"). Operationalized on Moodle, both structures lead to the same deployment (courses associated with students and resources). However, teachers mentioned their appreciation of being allowed to adapt the representation to their practices and perspectives.

4.5. Technical strategies for instantiating the table/tree model

Instantiating[1] the table/tree model with particular conceptual notions (in other words, defining a usable editor) may be addressed following two technical strategies: (1) by abstraction and then instantiation or (2) by technical adaptation.

The abstraction and then instantiation strategy (by-abstraction for short) consists in defining the tree nodes (in other words, the table items) as instantiable data structures. In such a case, defining an LSE consists in using an editing box to define the notions to use. We used the following data structure: an item (e.g., activity) is defined by a set of attributes, of which some are identical for all occurrences of the item (e.g., the attributes name and duration) and others are specific to the item's usage (e.g., the attributes beginning-date and ending-date); see supplementary material. The adaptation strategy consists, very simply, of adapting the code to use the relevant notions and data structures.

Using the by-abstraction strategy, the editor is straightforwardly usable and different variants may be offered: different sets of notions (e.g., the wiki and CSCL notions) may be defined as empty designs and presented as templates. Users select the pattern they want to use and design the scenario using the pattern's notions. The common/specific attributes data structure acts as a basic meta-model. As any meta-model, it allows for some representations and not for others. However, if more complex descriptions are needed, the data structure may be changed.

Using the adaptation strategy, ad hoc customizations may be made, including for the editor interfaces. For instance, the interfaces for defining items may be adapted according to the notions adopted and/or the context of use (e.g., presenting hints or learning design methodological considerations). In both cases, adaptations are basic because they do not affect the editor's functioning principles (the tree manipulation algorithms). The editor may also be customized in different ways, e.g., to upload student and resource information from a local database.

5. OBJECTIVE AND EVALUATION CRITERIA #1: PEDAGOGICAL EXPRESSIVENESS

The wiki and jigsaw examples show how a table allows for the distribution of tasks, roles, and resources, which is the basis of most scenarios. We mentioned that an Internet search showed that teachers often use tables to edit and prepare their sessions. To further evaluate the pedagogical expressiveness of the approach, we stuck to the basic table representation (i.e., without any enhancements) and considered the question, "Does a table representation allow for representation of a wide range of scenarios?" We considered this question in different ways.

The first evaluation action was conducted in the context of different usability tests (see Section 6), which involved K-12 teachers (plus one instructional designer and one computer scientist as not-targeted-users) and University lecturers. They were asked to edit and adapt two classical CSCL scenarios presented by a narrative text. All of them declared they could represent the scenario as they wished given their concerns. The fact that the CSCL notions provided were satisfactory is not surprising, as they have proven useful for representing a variety of collaborative scenarios. What this study showed is that the table was expressive enough for teachers to express their designs.

The second evaluation action was exploratory. We collected 25 CSCL scenarios from the research literature, all of which could be represented as tables (see supplementary material). The table representation, however, was more or less

[1] Instantiation of the table/tree model, which consists in adopting precise conceptual notions such as activity or topic, is not to be confused with the instantiation of a scenario pattern, which consists in defining how this pattern is used, e.g., instantiating jigsaw on insulation/heating techniques.

complete and more or less straightforward. First, the scenarios that included repetitions could not be represented as such, as they would be using a graph-based representation; some rows had to be duplicated. This is not an important issue, however, unless the number of repetitions is significant. Second, the basic representation did not allow for comprehensive representation of scenarios requiring run-time data from the enactment system, e.g., grouping students according to how they enact a given activity. This requires using an additional workflow representation (see Section 7.3) or enhancing the table with dynamic principles (see Section 7.2).

The third evaluation action was oriented towards practices. We considered the scenarios (referred to as "teaching strategy templates") designed by the large LAMS teacher community, such as role playing, problem-based learning, and predict-observe-explain (LAMS, 2013). Here again, most structures were simple sequences of tasks, and branching constructions could be represented using sub-rows. For instance, the role-playing general structure consists of a linear list of steps (e.g., considering documents), branching that leads different students to consider different activities (e.g., organizing a vote and making the ones that vote for and against consider different tasks), and then final activities with all the students. Such a structure could be represented as a table in the same form as the jigsaw in Figure 2. However, the LAMS representation is based on features related to operationalization: dynamic distribution of students based on the vote and scheduling implementation (students meet when they have all completed their branches). These features are native for LAMS, which addresses both representation and operationalization. As mentioned here above, they must be implemented as enhancements (see Sections 7.2 and 7.4, and supplementary material).

The fourth evaluation action considered equivalence with an LSE whose expressiveness had already been demonstrated. We considered SceDer, an LSE designed to create scenarios for one-to-one classrooms (Niramitranon et al., 2010). SceDer uses five conceptual notions (provider, what-to-do, receiver, resource and space) along with predefined values (e.g., teacher, question or file/picture). It has been evaluated as capable of modeling 13 high-priority educational scenarios. SceDer output contains an XML file that follows the interchange language COML for delivery on the enactment platform Group Scribbles. We showed that the table/tree model offered the same expressiveness by implementing a SceDer adaptation of the editor that offers these notions and predefined items (see supplementary material), and exports the design to COML (with exclusion of the Group Scribbles aspects, which were beyond the scope of this study).

These different elements suggest that tables (and, moreover, tables with enhancements) can be used to represent a wide range of scenarios and especially the classical examples that are often used.

6. Objective and Evaluation Criteria #2: Usability and Perceived Usefulness

To evaluate usability we examined the question, "Is the editor easy for the target users (teachers) to use?" Since the question of usability is related to matters currently being studied, we also considered teachers' perceived usefulness. In this section, we summarize the three usability tests conducted to date with a CSCL and an ad hoc instantiations.

The first test was conducted by the authors of this article. It included five K-12 teachers and the CSCL instantiation. The teachers were asked to achieve the prototypical task for which the table-approach was developed: edit and adapt two classical scenarios presented by a narrative text as they would to conduct the session in their classrooms. All teachers agreed or strongly agreed that they found the editor easy to use (see (Sobreira & Tchounikine, 2012) for details). They also all agreed or strongly agreed that it was useful for going into detail in the representation as suggested and made possible by the editor. All but one agreed or strongly agreed that the editor's features could be used to reflect on the scenario and gradually refine it, and would allow them to reflect on the enactment and orchestrate the session. The teacher who disagreed stated that the LSE was useful for structuring the session beforehand, but that she would not go back to it during the session; instead, she would revisit it after the session to reflect on what went well and what did not work, and to save such information data for future use/instantiation of the scenario.

The second test was conducted by another research team in another country (the GSIC group in Valladolid, Spain), in the context of a professional development workshop related to the use of ICT tools for teaching. Eighteen university lecturers used different LSEs, including ediT2 (CSCL instantiation). They were first asked to represent a jigsaw scenario presented in a narrative way and to answer a number of questions, including "[Do you think that] the editor is easy to use?" using a Likert scale 1-6 (6=totally agree). The average response was 4.9 (std. dev.=0.6, min. value=3). A second questionnaire administered after one week, during which participants had to edit scenarios at home, gave similar results. With respect to usefulness, the main feature mentioned by the lecturers was the possibility to automatically deploy the scenario on Moodle (which was offered via the GLUE!-PS middleware, see Section 7.4). The average responses to the items "[Do you think that] the editor is useful for reflection?" and "The editor is useful in practice." were 4.7 (std. dev.=0.4) and 4.1 (std. dev.=0.8), respectively, on the same Likert scale.

The third test was conducted with eight student teachers in pre-service education. Using the by-abstraction strategy, we instantiated the editor with notions and attributes taken from a standard preparation sheet (step, teacher's activity, student's activity, participant, and resource). The editor was presented as a possible means for preparing classroom sessions. All student teachers agreed it was easy to use. With respect to usefulness, seven perceived the table as a

pertinent way to deal with both a synthetic presentation (the table with the items, easy to reflect on and edit via the drag-and-drop function) and detailed presentation (the item's attributes and open text description, accessible by double-clicking on the item). Consistent with this perspective, they mentioned that such an editor would be useful to them if it were possible, once the design was edited, to print the overall representation and details on a sheet of paper in a way that would be practical for orchestrating the session in the classroom. Other perceived values included: checking that student activities planned as parallel sessions would not require the teacher's presence in different groups at the same time; proposing duration syntheses to help balance the teaching time across subject areas—mathematics, language, and so on—or types of activities. This again suggests the importance of making it easy to adapt the tool to local needs.

In all cases, only a few minutes were necessary to present the editor, which suggests intuitiveness and immediacy. These empirical data correlate with investigations of school teachers' ICT knowledge (Trimmer, 2006) and the value of tables as simple and effective representation devices (Nardi & Zarmer, 1993).

7. OBJECTIVE AND EVALUATION CRITERIA #3: COMPUTATIONAL EXPRESSIVENESS

The computational expressiveness of the table/tree pivotal model was evaluated by examining the question, "Under this approach, can the services requiring computation that can be found in LSEs (see Section 2) be offered?" This question was answered by showing that these services can be implemented.

7.1. Instantiation and constraint-checking support

Let us first consider the following services: offering scenario patterns, which is an approach receiving growing interest (Goodyear & Retalis, 2010; Hernández-Leo et al., 2010); instantiation support; and constraint-checking mechanisms (Pérez-Sanagustín et al., 2009). Combined, they make use cases such as the following possible: (1) generate a scenario from a pattern and the requested data; (2) as necessary, use the table interface to edit the scenario generated (e.g., move students from one group to another or change a random distribution of tasks according to students' characteristics); and (3) receive a warning if such editing conflicts with the pattern's intrinsic constraints (see supplementary material).

Within the table/tree approach, offering patterns and instantiation mechanisms consists of (1) creating an editing box for users to edit the data requested to instantiate a pattern and (2) creating a piece of code that generates the scenario model (the corresponding tree). Let us look at the wiki-based scenario example. An example of a pattern is: each of the n students is topic-editor for one topic (i.e., responsible for one synthesis text) and section-editor for one or more of the m sections making up the other topics. Instantiating this pattern is basic for small numbers but becomes a nightmare if, for example, n=21 and m=4. Using the pivotal model, this can be turned into a basic problem of generating a tree and distributing items (students, topics, and sections) across the tree's nodes.

Similarly, checking pattern-related constraints corresponds to analyses of the tree or sub-tree structure and/or items. Let us take the example of the jigsaw pattern constraints (Pérez-Sanagustín et al., 2009). Constraints such as "all script phases are performed by all participants" can be implemented by getting the participants-level items for each of the individual, expert, and jigsaw tree branches and checking the lists for correspondence. Constraints such as "in the individual phase, all participants receive the resource(s) related to one subject only" can be implemented by checking the subjects of the resource nodes in the branch of the appropriate phase. And so on.

Using a pivotal model also makes it possible to enhance the direct-manipulation features of the table with high-level meaningful modifications, e.g., for the wiki example, "move input from sections to topics."

7.2. Dynamic principles

To represent dynamic principles, e.g., grouping students according to their enactment of a previous task, the table's basic expressiveness may be enhanced by modifying the data structure of the node and/or attaching a specific piece of code to it. As examples: constructions such as "the union of groups G1 and G2" may be represented by offering a basic algebraic language; constructions such as "students who voted yes" require a piece of code that accesses the enactment framework database (e.g., Moodle) and gets the required data. This is ad hoc basic swift engineering work.

The implementation of dynamic principles illustrates again the interest of considering adaptability and computational flexibility (see Section 8). Attempting to offer an *a priori* set of general constructions covering all cases/all enactment platforms would be complex and hazardous. Offering a general framework (table/tree model and general management algorithms) that makes adaptations easy and costless makes it possible to offer LSEs that meet users' local needs.

7.3. Complex sequencing

A table representation does not allow for easy design of complex sequencings, which require graph-based modeling. As already mentioned, in many cases simple constructions are sufficient (Haake and Pfister, 2007) and a few rows can simply be duplicated. If complex sequencings are needed, however, it is possible to enhance the table editor with ad hoc means (e.g., using constructions based on numbering rows) but that would diminish the simplicity of the design. The general modeling principle according to which complex constructions should be addressed via different models or tools, offering different perspectives and advantages, suggests that adapted complementary means should be used instead.

To study this approach, we considered the interoperation of the table with the jBPM open-source workflow engine (jBPM, 2012). jBPM offers building blocks such as *split/join* nodes for branching and synchronization, and *for-each* nodes to represent repetitions. Implementing a piece of code transforming a tree representation into the jBPM format is basic engineering work. The output is an XML file that corresponds to a set of jBPM operation nodes with *split* and *join* connections. Such interoperation supports the following use case: (1) The scenario is sketched using the table representation. (2) The table representation is mapped (via the tree model) onto the jBPM concepts. (3) The resulting XML file is opened with the jBPM editor, which can be used to drag and drop *split* or *join* connectors between the activities and finalize the design (see supplementary material).

When a complementary workflow representation is used, different services can be offered, such as: automated configuration of an enactment framework (for instance, Harrer et al. (2009) have shown how a statechart can be used to configure the CeLS platform); Sobreira & Tchounikine, 2013); learning session monitoring assistance for teachers (Malzahn et al., 2008); guidance for students' enactment of the scenario via the workflow engine, e.g., prompting students with activities and/or making resources available according to the scenario scheduling (Haake and Pfister, 2007).

7.4. Operationalization

Within the approach proposed, operationalization is addressed by configuring an existing enactment framework via the tree pivotal model. There are two possible strategies: using middleware or developing an ad hoc mapping.

Middleware-based strategy (Moodle example)

The development of middleware that facilitates the deployment of learning designs on different operationalization frameworks is a desirable evolution in learning technologies, well in line with the approach presented in this article.

To explore this strategy, we considered the GLUE!-PS middleware (Prieto et al., 2011), which provides ready-to-use adaptors for different enactment frameworks, including Moodle. Considering the CSCL instantiation, we created an ediT2-to-GLUE!-PS adaptor. This makes the following use case possible: (1) Resources are created using Moodle, which offers specific editors to create links to online documents or many types of quizzes. (2) The editor is launched. The user clicks on "load resources" and references to the Moodle resources are uploaded into the editor's "resource" box (idem for participants). The user can then use editor features to edit the scenario. (3) The user clicks the export to GLUE!-PS button, opens the GLUE!-PS editor, imports this file, and selects Moodle as the target platform. (4) The adaptor generates a GLUE!-PS representation of the scenario (the representation can then be completed or adapted if necessary). (5) The user clicks the GLUE!-PS deployment button and the ready-to-use GLUE!-PSToMoodle adaptor creates a Moodle course and associates it with the corresponding resources and students. (6) When connecting to the LMS, the students are grouped and offered resources according to the scenario designed via ediT2 (see supplementary material). The version that was tested with university lecturers used this interoperation, with the GLUE!-PS step hidden so as to not confuse users.

An advantage of middleware is that it makes it possible to deploy the editor on different platforms. Moreover, technical modifications of the platform only impact the corresponding adaptor. A drawback is that the mapping is generic and cannot be customized.

Ad hoc strategy (MediaWiki example)

The ad hoc implementation strategy consists in implementing a specific piece of code that directly matches the table structure and the targeted enactment framework. This is the strategy we adopted for the wiki example, using MediaWiki (2013) as the target enactment framework. Once the editor/MediaWiki notion were matched, a simple parser was used to analyze the tree (i.e., in this example, the topics and topic sections) and call the corresponding MediaWiki API feature (e.g., ready-to-use services such as "edit/create a page"); see supplementary material.

A key advantage of the ad hoc strategy is that it supports LSE customization. As shown in Figure 2, one can easily define 'an LSE for designing wiki-based scenarios and operationalizing them on MediaWiki', customized to allow the use of meaningful notions (e.g., topic or editor) and offering the services that can be expected in this context (e.g., managing the list of students only once). Another advantage is linking representation and operationalization aspects to address expressiveness and dynamic principle issues (see sections 4.4 and 7.2). For instance, ad hoc operationalization makes it possible to link some phases (e.g., a grouping phase) with specific features (e.g., a voting tool) to implement dynamic grouping as LAMS does. It also makes it possible to control the generation of activities on the enactment platform, i.e., represent synchronicity features such as progressing to the next step if and only if all students or all groups are done with a task. An ad hoc process also limits the information loss caused by successive mappings between the data models of the tools involved in each step of a middleware-based process (Muñoz-Cristóbal et al., 2012).

The disadvantage of the ad hoc strategy is that it leads to locally re-inventing code that has already been produced elsewhere. Its technical complexity depends on the target enactment framework API and/or technical documentation.

8. OBJECTIVE AND EVALUATION CRITERIA #4: COMPUTATIONAL FLEXIBILITY

The computational flexibility of the table/tree model was evaluated by examining the question, "Can the pivotal model approach be used to easily adapt the editor to local demands or constraints and easily extend it to offer additional services?" In previous sections we showed this could be done; the question here is whether it is easy. This is crucial, as the overall idea is to adapt the table to local needs and offer additional services if/when needed.

From a computational flexibility perspective, the interest of a pivotal model approach is to natively consider adaptation and extension. A tree is adopted because it is a classic and simple data structure, to which manipulations (generation, transformation, and analysis) can be implemented by simple, uniform—and thus reusable—algorithms. We show in this section that adapting the table to local needs and offering additional services is simple by presenting the implementation principles (see also the supplementary material). Although this is indeed technical, it allows computer scientists to note that these processes are simple. We also present a pseudo-evaluation.

Table implementation and adaptation

The table's basic representation features (e.g., displacing items between cells or creating rows) correspond to trivial algorithms (displacing items between nodes or creating branches). Features such as splitting/merging nodes (i.e., creating/merging sub-branches) or moving a column while maintaining consistency between the table representation and the tree model require propagating the modifications via tree manipulation algorithms, which are less trivial but of standard practice. These aspects form the architectural foundation; they do not need to be adapted.

Creating a particular instantiation is straightforward if using the by-abstraction strategy. The adaptation strategy requires basic engineering work. As an example, the wiki version required adapting the data structure to introduce the topic (etc.) notions, implementing the load-students-from-MediaWiki feature and the operationalization service (configuration of MediaWiki from the scenario), and adapting the table's top-level menu (access to additional services). This took less than 4 days of engineering work, half of which was spend on conceptual mapping.

The table interface may be implemented in a number of ways. We used Adobe® Flash and AIR® technologies (AIR, 2012) for the basic and enhanced editors, respectively. The basic editor can therefore be opened from a Web browser.

Additional services

We showed in Section 7 that instantiation support and constraint-checking services correspond to basic tree manipulations. This is made particularly easy when using a lambda-calculus language; we used Haskell (2012). Mapping to other architectures (enactment frameworks, workflows, or middleware) is a more or less difficult engineering task, depending on the target system's complexity and documentation. For instance, applying the ad hoc operationalization strategy to MediaWiki was easy, but turned out to require more difficult engineering work for Moodle due to the enactment framework structure, services, and documentation.

From a computational flexibility perspective, an advantage of using a pivotal model is the use of internal transformations. For instance, mapping the editor with a given target (e.g., GLUE!-PS) is technically made easier when focusing on a given representation structure (i.e., ordered list of notions). However, if offered representation flexibility, teachers may change this order. This may be smartly addressed by developing one mapping, based on a canonical structure, and using a piece of code to transform a scenario representation from the structure used by the teacher to the canonical structure. Such a strategy facilitates the implementation of complex services. Technically, changing a representation structure is not trivial but it is uniform.

A pseudo-evaluation

It is not possible to create the conditions for evaluating characteristics such as computational flexibility by empirical comparison, e.g., comparing how long it takes to create a service from one software package or another. This is why we explained the principles and gave examples to facilitate conclusive analyses. However, to evaluate computational flexibility, we also studied adaptation by an external actor. The opportunity came from collaboration with the GSIC research group, which performs research related to monitoring CSCL scenarios. To get input from the monitoring device, at design time teachers must provide information related to the activities (e.g., the deadlines, the social level of the activity, whether student participation is optional or mandatory, and whether the activity is remote, face-to-face, or blended) and identify the ICT tools that support such activities, e.g. expected use of the resource and types of interaction the teacher wants to monitor at run-time (Rodríguez-Triana et al., 2013). Since this monitoring information could not be edited in their standard editor, the group was interested in using ediT2.

Adapting the editor to include representation of the monitoring data required different actions: (1) defining item data structures to include the requested information; (2) enabling users to edit these data when building a design; (3) implementing *if/then* controls and hints, i.e., pieces of code that are executed when the teacher edits the corresponding data and provide methodological input; and (4) modifying the "save" feature to export the design (the scenario) in the format requested by the monitoring device. These changes were processed by a member of the GSIC group ("the adapter") as someone who had not participated in the editor design or implementation in any way. The adapter managed

aspects (1) and (2) using the by-abstraction strategy, i.e., she defined an empty design featuring the different attributes that were required. This was straightforward (about 1 hour). To implement the controls (3), the adapter was given the editor code and a short presentation (about 1 hour). She considered controls/hints such as "*If* the type of resource is A, *then* the actions that can be monitored are B" and "*If* participation in the activity is mandatory and there are not enough actions that can be monitored to ensure student participation, *then* report the lack of data." Implementing such controls required about 1-2 hours each. In a second version, she decided to modify the code again to check all controls at the same time after the editing, which was again basic. Editing the export format (4) took about 2 hours. In other words, adapting the editor to these specific needs was easy and of low engineering cost.

9. COMPARISON WITH OTHER WORKS AND DISCUSSION

9.1. Comparison with other methodological approaches

The design rationale for the proposed approach was discussed in Section 3. Let us come back to it now to clarify three important and original considerations.

In Learning Design, the canonical approach is to identify what types of scenario should be representable and implement an editor based on these requirements. Building on the decision to use table interfaces may thus be seen as a techno-centric approach. This is not the case: although unusual, this strategy is based on another type of requirement, namely user-centric requirements. Research works related to LSEs and Learning Design must address how to represent scenarios in the best way and how to design means that can/will be used by the target users. Unfortunately, designing tools that provide educational value is not sufficient for these tools to be adopted by teachers (Tchounikine, 2011; Demetriadis et al., 2003). Criteria for adoption include user-friendliness and easy transition from existing practices and tools. This is the rationale for considering devices that teachers know and use. Within their analysis of attitudes underlying acceptance of/resistance to the infusion of technology into schools, Demetriadis and his colleagues (2003) suggest the inspiring perspective of "cultures in negotiation." Offering flexible tools that can be adapted by local institutions (see next paragraph), is a way to support some of the technological aspects of this negotiation.

With respect to genericity/adaptability, we have shown that the approach we propose may be used in different ways: to offer a range of editors that propose different conceptual notions using the by-abstraction strategy, and to offer an editor that can be customized by modifying the code at low engineering cost. With respect to this latter option, it can be argued that systems should be usable off-the-shelf because teachers cannot be expected to technically adapt systems. This suggests offering generic architecture/languages. However, while generic means are powerful, users such as teachers may not want to reflect in generic terms, despite being able to do so. In some cases, there may be good reasons to impose particular specific or generic notions, e.g., industrialization or professional training. When this is not the case, there is no particular reason to ask teachers using wiki-based scenarios to fight with a generic system if they can be offered, at low cost, specific meaningful modeling means (e.g., topic and editor notions) and features adapted to their local needs (e.g., direct upload of students from the local database and straightforward deployment to the wiki used). When considering adoption, computational flexibility (easy adaptation to the local technical and/or social ecosystem) is very important: a few hours of engineering work may change an odd tool into an adapted means.

The underlying perspective is that it is pertinent, at least in some cases, to adapt tools to users and contexts. Here again, one may take another position and argue that LSEs should reify good instructional practices and that teachers should be trained to use them. However, this is a theoretical position, and in many cases not practical. Moreover, this approach may also be questioned as such. One may consider that teachers, as the actors in charge of the actual settings and as actors with professional practices, should be able to use tools in line with their practices and perspectives. These questions do not have one single answer, they must be considered within a holistic perspective that includes institutional and practical aspects. Depending on the country and context, teachers may edit existing materials (e.g., scenarios) only or they may have more latitude (or fewer resources) and design materials themselves; they may or may not work within a specific methodology; they may or may not receive specific training; they may or may not have technical support; and so on. Acknowledging this diversity requires offering different LSEs with different characteristics.

9.2. Comparison with other systems

With respect to representation, many LSEs present end users with graph-based interfaces (Botturi and Stubbs, 2007), especially when considering complex sequencing, and build on the top of workflow engines (e.g., Flex-eL (Sadiq et al., 2002), (Haake and Pfister, 2007), MoCoLADe (Harrer et al., 2007), and LeadFlow4LD (Palomino-Ramírez et al., 2008)). LSEs related to XML-based representations (e.g., Learning Designer) usually build on hierarchical representations. Table and graph representations are different options with different characteristics. We have shown how a table may be complemented by a graph-representation. This makes sense when attempting to benefit from both table simplicity and graph expressiveness. However, users who require complex representations and/or are at ease with graph-representation should use editors that are natively based on graph representations, such as MoCoLADe, and thereby benefit from the specific associated features. One may argue that graph-based editors can be used to represent

both basic and complex scenarios. This is true; however, with respect to adoption and smooth transitions, presenting practitioners with complex interfaces for the sake of providing features that may be used only rarely is, in our opinion, exactly what should be avoided. More (comprehensive languages, all-in-one architecture offering a full range of services) is not necessarily better, as "more" often comes with complexity and reduced flexibility.

With respect to operationalization, an important advantage of using a pivotal model is that different platforms can be considered. This is also the rationale for the GLUE!-PS middleware. This approach is to be contrasted with languages linked to enactment platforms such as LAMS (LAMS, 2012) or CeLS (Ronen et al., 2006), within which scenarios are, by definition, directly operationalized but representation and editing means have to comply with enactment platform computational constraints. Using middleware is possible when the conceptual frameworks match; from this perspective, GLUE!-PS offers a fairly general framework. The ad hoc implementation strategy is more flexible and allows specific features to be implemented. For instance, in the specific MediaWiki implementation, we were able to restrict editing to the students identified as editors.

With respect to software engineering, our use of a pivotal model may be seen as a particular (simple) case of meta-modeling. The way we used the tree model to implement different versions of LSEs is similar to the way MoCoLADe is based on a generic graph-based modeler. From a technical perspective, an alternative to a tree model could be directed acyclic graphs, which would allow constructions that require some duplication in a tree model. We opted for trees to stick to the simplicity principle of the approach.

If one refers to the classification framework proposed in (Botturi et al., 2006), a table/tree representation addresses a conceptual level of elaboration (a general, aggregate view of the design, indicating its main elements); it offers flat stratification and a single perspective (one view of the entities) via a visual notation system (the table).

9.3. Discussion: Fostering effective usage of modeling tools

Many studies have been conducted to analyze and compare how different approaches to Learning Design and different modeling tools supported the design, edition, and/or implementation of given scenarios, e.g., recently, (Persico et al., 2013) and (Prieto et al., 2013). These studies actually confirm the usefulness of each particular new tool/approach. Taking a different perspective, a specific study was conducted to analyze how a group of teachers perceived and used the table-based editor's high flexibility and Collage's pattern-based guidance (Prieto et al., 2014). Here again, the data suggests that there is no single silver bullet: teachers appreciate different kinds of support depending on the moment and the concrete task at hand. This finding correlates with other studies reported in the literature; see (Prieto et al., 2014).

With respect to promoting the practice of Learning Design, these studies suggest that progress will not be made by designing *the* LSE that will solve all issues for all users but by offering a wide set of possibilities that users, especially teachers, may choose according to their perspectives, needs, or context. This is the rationale for the approach presented in this article. Rather than attempting to design yet another general editor, our approach enriches the technical offering with an editor that has intrinsic limitations but offers specific characteristics (simplicity, flexibility, adaptability), and can be used when sufficient and convenient. This is meant to be positioned next to other proposals offering other characteristics, such as LSEs covering a wide spectrum of scenarios (e.g., LSEs based on IMS-LD), LSEs dedicated to a specific type of scenario (e.g., MoCoLADe for CSCL) or offering specific guidance (e.g., Collage), or LSEs based on or related to specific enactment frameworks (e.g., LAMS).

Offering a wide spectrum of tools, however, is not enough. There is also a need to support teachers in going from their pedagogical intentions and usual practices to modeling and operationalizing their learning designs (and, throughout this process, adopting the tools that best fit their needs). An interesting approach to this issue is studied in the METIS project (METIS, 2014): acknowledging that offering different tools is of interest; developing technology to integrate these different tools (Hernández-Leo et al., 2014); and, of core importance, organizing different types of professional development workshops, including some that address authentic learning situations. From this perspective, the approach we propose presents three specific interests: (1) It provides a representation device that is easy to use and may facilitate a smooth transition to more advanced means. (2) Building on the by-abstraction adaptation strategy and representation flexibility, it offers the means for teachers to test different representation structures. (3) On the technical side, using the pivotal model, it facilitates interoperation with other systems.

10. CONCLUSIONS

We have presented an innovative approach to the design of modeling editors for end users who are not specifically skilled or trained, and whose concern is not to become modeling specialists. For such cases, effective adoption depends on software aspects and on many other considerations (e.g., social, professional, and institutional factors). We have shown that a table-based approach can be used to fulfill the software requirements of designing a system that is simple to use and flexible; of limited expressiveness but able to support representation of a wide range of scenarios; that can offer complex services as add-ons; and, last but not least, that is easy to adapt.

The proposed approach can be used to offer users systems that can be adapted locally to their needs and/or practices,

no more complex than needed, and easy to interoperate with other systems if useful. The computational flexibility provided by the pivotal model approach may also be used to study, and adapt the system to, perceived/effective usefulness and users' actual usages. Similarly, with respect to defining LSEs that meet the needs and requests of a group of users (and, to some extent, to come close to participative design), the by-abstraction adaptation strategy may be used to explore local instantiations with informant practitioners. Finally, it may also be used, when conducting professional development workshops, to test/use different representations systems at low cost.

11. REFERENCES

AIR, 2012. www.adobe.com/products/air.html (accessed Dec. 2012)

Botturi, L., Derntl, M., Boot, E., Figl, K. (2006). A Classification Framework for Educational Modeling Languages in Instructional Design. In: Proceedings of 6th IEEE International Conference on Advanced Learning Technologies, pp. 1216-1220.

Botturi, L., Stubbs, T. (2007). Handbook of Visual Languages for Instructional Design: Theory and Practices. IGI, Hershey.

Coppercore (2012). The IMS Learning Design Engine. Available at http://coppercore.sourceforge.net (accessed Dec. 2012).

Demetriadis, S., Barbas, A., Molohides, A., Palaigeorgiou, G., Psillos, D., Vlahavas, I., Tsoukalas, I., & Pombortsis, A. (2003). Cultures in Negotiation: Teachers' Acceptance/Resistance Attitudes Considering the Infusion of Technology into Schools. Computers & Education, 41(1), 19-37.

Derntl, M., Neumann, S., Griffiths D., Oberhuemer, P. (2011). The conceptual structure of IMS Learning Design does not impede its use for authoring. IEEE Transactions on Learning Technologies 5(1), 74-86.

Dillenbourg, P., Dimitriadis, Y., Nussbaum, M. (2013). Design for Classroom Orchestration. Computers & Education.

Dillenbourg, P., Jermann, P. (2010). Technology for classroom orchestration. In: Khine, I. M. (Ed.), The New Science of Learning: Computers, Cognition and Collaboration in Education, Springer, pp. 525-552.

Dillenbourg, P., Hong, F. (2008). The mechanics of CSCL macro scripts. International Journal of Computer Supported Collaborative Learning 3(1), 5-23.

Dillenbourg, P., Tchounikine, P. (2007). Flexibility in macro-scripts for CSCL. Journal of Computer Assisted Learning 23(1), 1-13.

Goodyear, P., Retalis, S. (2010). Technology-enhanced learning: design patterns and pattern languages. Sense Publishers, Rotterdam.

Haake, J.M., Pfister, H.-R. (2007). Flexible scripting in Net-Based Learning Groups. In: Fischer, F., Kollar, I., Mandl, H., Haake, J. M. (Eds.), CSCL Series, Springer, pp.155-175.

Harrer, A., Kohen-Vacs, D., Roth, B., Malzahn, N., Hoppe, U., Ronen, M. (2009). Design and Enactment of Collaboration Scripts – An integrative approach with graphical notations and learning platforms. In: Proceedings of the 9th International Conference on Computer Supported Collaborative Learning, pp. 198-200.

Harrer, A., Malzahn, N., Hoppe, H. U. (2007). Graphical Modeling and Simulation of Learning Designs. In: Hirashima, T., Hoppe, H. U., Young, S. S. (Eds.), Supporting Learning Flow through Integrative Technologies. IOS Press, Amsterdam, pp. 291-294.

Haskell (2012). www.haskell.org (accessed December 2012).

Hernández-Leo, D., Asensio-Pérez, J.I., Derntl, M., Prieto, L.P., Chacón, J., (2014). ILDE: Community Environment for Conceptualizing, Authoring and Deploying Learning Activities. In: European Conference on Technology Enhanced Learning, Graz.

Hernández-Leo, D., Asensio-Pérez, J. I., Dimitriadis, Y., Villasclaras-Fernández, E. D. (2010). Generating CSCL Scripts: From a Conceptual Model of Pattern Languages to the Design a real situation. In: Goodyear, P., Retalis, S. (Eds.), Technology-enhanced learning: design patterns and pattern languages, Sense Publishers (appendix).

Hernández-Leo, D., Villasclaras-Fernández, E. D., Asensio-Pérez, J. I., Dimitriadis, Y., Jorrín-Abellán, I. M., Ruiz-Requies, I., Rubia-Avi, B. (2006). COLLAGE: A collaborative Learning Design editor based on patterns. Journal of Educational Technology & Society 9(1), 58-71.

IMS-LD (2013). www.imsglobal.org (accessed Aug. 2013).

jBPM (2012). www.jboss.org/jbpm (accessed Dec. 2012).

Kobbe, L., Weinberger, A., Dillenbourg, P., Harrer, A., Hämäläinen, R., Häkkinen, P., Fischer, F. (2007). Specifying Computer-Supported Collaboration Scripts. International Journal of Computer Supported Collaborative Learning 2(2-3), 211-224.

LAMS (2013). implementinglearningdesigns.lamsfoundation.org (accessed Oct. 2013).

LAMS (2012). www.lamsfoundation.org (accessed Aug. 2012).

Larnaca (2012). The Larnaca Declaration on Learning Design. Available at http://www.larnacadeclaration.org (accessed Dec. 2012).

Laurillard, D., Charlton, P., Craft, B., Dimakopoulos, D., Ljubojevic, D., Magoulas, G., Masterman, E., Pujadas, R., Whitley, E.A., Whittlestone, K. (2013). A constructionist learning environment for teachers to model learning designs. Journal of Computer Assisted Learning 29(1), 15-30.

Lejeune, A., Ney, M., Weinberger, A., Pedaste, M., Bollen, L., Hovardas, T., Hoppe, U., de Jong, T. (2009). Learning Activity Spaces: Towards flexibility in learning design? In: International Conference on Advanced Learning Technologies, pp. 433-437.

Malzahn, N., Pokrandt, M., Hoppe, H. U. (2008). Extending a Learning Design Editor with a Monitoring Component. In: International Conference on Computers in Education, pp. 499-504.

MediaWiki (2013). www.mediawiki.org (accessed Mar. 2013).

METIS (2014). www.metis-project.org (accessed July 2014).

Moodle (2012). www.moodle.org (accessed Aug. 2012).

Mor, Y., Craft, B. (2012). Learning design: reflections upon the current landscape. Research in Learning Technology 20.

Muñoz-Cristóbal, J.A., Prieto, L.P., Asensio-Pérez, J.I., Jorrín-Abellán, I.M., Dimitriadis, Y. (2012). Lost in Translation from Abstract Learning Design to ICT Implementation: A Study Using Moodle for CSCL. In: Ravenscroft, A., Lindstaedt S., Kloos, C. Hernández-Leo, D. (eds), 21st Century Learning for 21st Century Skills, pp 264-277, Springer Berlin Heidelberg.

Nardi, B.A. & Zarmer, C.L. (1993). Beyond Models and Metaphors: Visual Formalism in User Interface Design. Journal of Visual Languages and Computing 4, 5-33.

Neumann, S., Klebl, M., Griffiths, D., Hernández-Leo, D., de la FuenteValentín, L., Hummel, H., Brouns, F., Derntl, M., Oberhuemer, P. (2010). Report of the Results of an IMS Learning Design Expert Workshop. International Journal of Emerging Technologies in Learning 5(1), 58-72.

Niramitranon, J, Sharples, M. & Greenhalgh, C. (2010). Orchestrating Learning in a one-to-one Technology Classroom. In Khine, M. S. & and Saleh, I. M. (eds.) New Science of Learning: Cognition, Computers and Collaboration in Education. Springer, pp. 451-468.

Palomino-Ramírez, L., Bote-Lorenzo, M., Asensio-Pérez, J., Dimitriadis, Y. (2008). LeadFlow4LD: Learning and Data Flow Composition-Based Solution for Learning Design in CSCL. In: Groupware: Design, Implementation, and Use, v. 5411, pp. 266-280.

Pérez-Sanagustín, M., Burgos, J., Hernández-Leo, D., Blat, J. (2009). Considering the intrinsic constraints for groups management of TAPPS & Jigsaw CLFPs. In: International Conference on Intelligent Networking and Collaborative Systems, 317-322.

Persico, D., Pozzi, F., Anastopoulou, S., Conole, G., Craft, B., Dimitriadis, Y., Hernández-Leo, D., Kali, Y., Mor, Y., Pérez.Sanagustín, M., Walmsley, H. (2013). Learning design Rashomon I - supporting the design of one lesson through different approaches. Journal of Research in Learning Technologies, 21.

Prieto, L. P., Tchounikine, P., Asensio-Perez, J. I., Sobreira, P., & Dimitriadis, Y. (2014). Exploring teachers' perceptions on different CSCL script editing tools. Computers & Education, 78, 383-396.

Prieto, L. P., Dimitriadis, Y., Craft, B., Derntl, M., Emin, V., Katsamani, M., Laurillard, D., Masterman, E., Retalis, S., & Villasclaras, E. (2013). Learning design Rashomon II: exploring one lesson through multiple tools. Research in Learning Technology, 21.

Prieto, L. P., Asensio-Pérez, J. I., Dimitriadis, Y. A., Gómez-Sánchez, E., Muñoz-Cristóbal, J. A. (2011). GLUE!-PS: A Multi-language Architecture and Data Model to Deploy TEL Designs to Multiple Learning Environments. In: European Conference on Technology Enhanced Learning, pp. 285-298.

ReCourse (2013). http://tencompetence-project.bolton.ac.uk/ldauthor/ (accessed Sept. 2013).

Rodríguez-Triana, M.J., Martínez-Monés, A., Asensio-Pérez, J.I., Dimitriadis, Y., A. (2013). Towards a Script-Aware Monitoring Process of Computer-Supported Collaborative Learning Scenarios. International Journal on Technology Enhanced Learning, special issue on Learning Analytics, 5(2), 151-167.

Ronen, M., Kohen-Vacs, D., Raz-Fogel, N. (2006). Adopt & Adapt: Structuring, Sharing and Reusing Asynchronous Collaborative Pedagogy. In: Proceedings of the 7th International Conference of the Learning Sciences, pp. 599-605.

Sadiq, S. W., Sadiq, W. & Orlowska, M. E. (2002). Workflow driven e-learning beyond collaborative environments. In: International NAISO Congress on Networked Learning in a Global Environment, Challenges and Solutions for Virtual Education, pp. 1-7.

Sobreira, P., & Tchounikine. (2012). A model for flexibly editing CSCL scripts. International Journal of Computer-Supported Collaborative Learning, 7(4), 567-592.

Sobreira, P., & Tchounikine, P. (2013). CSCL scripts: interoperating table and graph representations. In International Conference on Computer Supported Collaborative Learning, Madison e USA (Vol. 2, pp. 165-168).

Tchounikine, P. (2013). Clarifying Design for Orchestration: Orchestration and Orchestrable Technology, Scripting and Conducting. Computers and Education 69, 500-503.

Tchounikine, P. (2011). Computer Science and Educational Software Design – A Resource for Multidisciplinary Work in Technology Enhanced Learning. Springer. DOI 10.1007/978-3-642-20003-8_6.

Tchounikine P., Mørch A.I., Bannon L. (2009). A computer science perspective on TEL research. In: N. Balacheff, S. Ludvigsen, T. de Jong, A. Lazonder, S. Barnes (Eds) Technology-Enhanced Learning – Principles and Products. Springer.

Trimmer, K. (2006). Teacher ICT Skills: Evaluation of the Information and Communication Technology Knowledge and Skill Levels of Western Australian Government School Teachers. Australian Evaluation Society International Conference, Darwin, Australia (retrieved from http://www.aes.asn.au, April 2012).

Villasclaras-Fernández, E. D., Hernández-Gonzalo, J. A., Hernández-Leo, D., Asensio-Pérez, J. I., Dimitriadis, Y., Martínez-Monés, A. (2009). InstanceCollage: a tool for the particularization of collaborative IMS-LD scripts. Journal of Educational Technology & Society 12 (3), 56–70.

Weinberger, A., Kollar, I., Dimitriadis, Y., Mäkitalo-Siegl, K., Fischer, F. (2009). Computer-supported collaboration scripts: Theory and practice of scripting CSCL. In: N. Balacheff, S. Ludvigsen, T. de Jong, A. Lazonder, S. Barnes (Eds) Technology-Enhanced Learning – Principles and Products. Springer.

Williams, D. L., Boone, R., Kingsley, K. V. (2004). Teacher beliefs about educational software: A Delphi study. Journal of Research on Technology in Education 36(3), 213-229.